PSis - Programação de Sistemas - 2018/2019
Exame 1ª Época, 12 de Junho de 2018, 18h30, Duração: 2h

Software Architectures [6.0]

1. [1.0 points]

- What is an **Architectural Patterns**?

is a set of principles—a coarse grained pattern that provides an abstract framework for a family of systems

2. [1.0 points]

- Describe two advantages of using **Architectural Patterns** when implementing a complex system?

Provides a set of components and characteristics well known
Provides a well known solution to a problem
allows easier communication between members

3. [1.0 points]

Most of the project implementations (distributed concentration game) did not implemented a pure **client-server Architectural pattern**.

- Describe the **client-server Architectural pattern**
- Explain how the project implementation differed from this pattern (If you group implemented a pure client-server in the project describe its use in the project).

Client/Server – client makes request → Server answers
In the project this did not happened. The client would send a message that would not receive directly a response.
The server would send a new message to all the clients as a reaction to a message. A client could receive a lot of messages from the server without sending a request.

4. [1.0 points]

- Describe the **Layered Architecture pattern.**
- Describe how this architectural pattern was applied in the project.
- Present and describe two advantages of its use.

Grouping of related functionalities within an application into distinct layers.
Functionality within each layer is related by a common role or responsibility.
A layered system is organized hierarchically, each layer providing service to the layer above it and serving as a client to the layer below.

UI, communication, game rules, game storage

- Communication between layers is explicit
- Supports reuse, Isolation, and design based on increasing levels of abstraction

5. [1.0 points] In the project, the communication between the client and the server was performed using low level socket messages, but RPC could also be used.

- Describe what is RPC
- Give an example of how the interface between the server and client would be defined.

Component functionality is provided as a set of function calls
Applications call methods/function that are executed by a remote node

Void PickCard(x, y)
Void turnCard(x, y, "aa", UP)
...

Testing / software correctness [5.0]

6. [1.0 points]

- What is **integration testing**?
- What class of errors this tests try to find?
- How this type of tests gains from the use of a layered architecture in a system?

Integration testing is a systematic technique for constructing the program structure.
Tries to find errors that come from incompatibilities of the integrated modules or errors that only occur when a module is used in conjunction with another.
Since layers are independent of each other the integration of these layers is easier, with a well-defined set of flows it is easy to perform the integration tests.

Since the functionalities are well seprarated the integration and tes is easier

7. [2.0 points] The teaching staff provided a set of functions to help the development of the project. One of such functions was the **void init_board(int dim)** function that would create a square board and fill it with the pairs of letters.

- Present **3 unitary tests** that could be applied to that module
- for each test
    - Describe the error that would be found,
    - Present the pseudo-code of the test
    - Describe how to validate the result after performing the test

Create a board with 4 rows/columns, used to verify if the board is filled correctly or not, verify all boad places
Call the function with a negative value, to verify is this input is validated, verify the value of the global variable is NULL
Call the function with and odd value (), to verify is this input is validated, verify the value of the global

variable is NULL

8. [1.0 points] Security testing guarantees that a system is capable of handling attacks. The project could suffer from different attacks from malicious clients (for instance a BOT with bad behavior).

- Describe two possible security tests to guarantee the project security .
- For each test
  - identify the possible attack
  - what needed to be coded in the project to pass such test.

Verify the data on all received messages
Correctly implement all game rules

9. [1.0 points]

- Describe why sending a 32 bits integers through a INET socket without any transformation (as presented in the example) affects the compatibility of a system:

```
int n;
…
write(sock_fd, &n, sizeof(n));
```

- Describe what changes in the previous code should be made to guarantee compatibility between components that communicate through that socket

Different computers use processors that store the integers in different ways (big vs little endian) if the integer are sent as is, the received bits correspond to a different value that was sent.
The sender should transform the integer to a canonical value (e.g. using the htonl) and the receiver should transform the received value to the internal form

Processes / Threads / IPC [4.0]

10. [1.0 point]

- Explain how unix **signals** are implemented and how they may be used in the context of a complex system.

The process call the kill system call, the kernel redirect the signal to the suitable process and select the signal handler of that process to be executed. Process on continues executions, process 2 is interrupted and the signal handler starts executing

11. Most of the project implementations used stream sockets, where each socket was processed by a different thread, and had the board shared between all those threads so that each thread could verify independently the status of the cards (UP, DOWN, …).

11.a. [1.0 points]
Still maintaining each thread processing one socket (one socket <-> one thread):

- Describe a different organization of the server so that only one thread would access de board (the board was a local variable on that thread and no other thread could access it)
- Describe the different processing nodes and communication mechanism of this new approach.

If necessary draw a schema of the server organization (representing the various threads and communication channels.

There were a series if pipes that allowes the “socket threads”to send requests to the board thread

11.b. [1.0 points]

- Present and describe one advantage and one drawback of this new server organizations?

Advantage – no synchronization
Drawback – lack of concurrency – speed.

12. [1.0 point]

- Compare **pipes** and **unix domain stream sockets** with respect to:
  - Addresses
  - Communication isolation.
- Explain in detail the similarities or differences presented.

Addresses –
pipes do not have addresses, sockets use file names

isolation
any child process can use the same pipe to communicate
socket only the 2 processes that established connection can communicate

Synchronization [6.0]

14. [2.0 point] Suppose that a thread code has a bug and that thread dies inside a critical region before unlocking the a locked mutex.

- Describe what happens to the mutex in such case.
- Describe what happens to the system in such case.

If the mutex is regular it will continue blocked, special mutexes, can get unblocked in this case

The system will mostly not work correctly:
Either deadlock
Or inconsistencies in the data

14. [2.0 point] Condition variables are a type of synchronization objects defined in POSIX.

- Describe when they should be used.
- Describe how they are used (with a pseudo-code example)

They should be use when it is necessary to leave a c ritical region in the midle and the return to the same pint (block inside a critical region)
Or notify a wating thread of an event.

A thre acquires a lock and then blocks waiting for the condition variable. The mutex is released.

Another thread signals the condition variable and the waiting thread starts executing by first acquiring the mutex.

15. [2.0 point] In the project it was necessary to store the players in a list. Every time a new client connected, a new structure would be inserted into the list, and every time a board update was made, it would be forwarded to every player.
The following code presents a pseudo-code implementation of the two functions **insertPlayer** and **sendBoardUpdate**.

insertPlayer	**sendBoardUpdate**

newPlayer = initializePlayer()
 	
newPlayer-> next = playerList;
playerList = newPlayer;
	aux = playerList;
	while(aux != NULL){
	   sendUpdate(aux, x, y, UP)
	   aux = aux -> next;
	}

Supposing that the players were never removed from the list:

- Identify the race condition
- Describe what are the effects when the race condition happens.
- Change the code so that the critical region is correctly guarded.

If the insertPlayer is executed concurrentky, the race condition occurer in the

newPlayer-> next = playerList;
playerList = newPlayer;

instructions

if two threads execute this code “simultaneous” the consistency of the system is not guaranteed (one of the inserts may be oberrun

theses intructions should be guarded by a mutex lock/unlock